CHAPTER 3

WEB PAGES REPRESENTATION

This chapter deals with transforming the web pages to resource description framework and also discuss about the advantages and disadvantages of conversion and problems that were identified and resolved.

3.1 INTRODUCTION

The original web mainly concentrates on the interchange of documents whereas the semantic web is about common formats for integration and combination of data from diverse sources to carry out sophisticated tasks for users. The web and Hyper Text Markup Language (HTML) gives access to information, but it is too narrow in scope to handle the many tasks since it cannot accommodate every type of data. Web hypertext application group (2010) affirms about HTML and its need. XML enables to create richly structured documents thereby provide machine interoperability. But it is difficult to merge XML files from two different applications. To take a (XML) query on an XML document, and add in some constraints from another document and merging the two queries is complicated. Thus there is a need to adopt a relational model. XML gives us the format to record domain-neutral data, but RDF gives us the methodology to record complete domain-neutral statements.

The Resource Description Framework enables the creation and exchange of resource metadata as any other Web data. RDF is a technique to
record statements about resources so that machines can easily pick up the
statements. RDF allows one set of statements to be merged with another set of
statements, even though the information contained in each set of statements
may differ dramatically. RDF properties make it easy to merge data and query
across data from different sources. RDF can be used in a resource discovery to
provide better search engine capabilities. Bernard et al (2006) discuss about the
automatic information gathering by search engines. Mohammed Kayed and
Chia-Hui Chang (2009) describe about the content extraction from template
pages. Web document retrieval is discussed by Cho (2007). The concept of
semantic web enables search engine and browsers to provide more relevance in
search results.

3.2 OVERALL ARCHITECTURE OF THE SYSTEM

The proposed system is aimed at providing automated information
access based on machine understandable semantics of data thereby providing
qualitatively new level of service. i.e instead of long list of links displayed on
the page, a collective content retrieved from multiple pages is presented to the
user. For any given query, the system tries to fetch meaningful sentences from
every page that matches the query, combines it and displays that as a result.

It customizes the search process. Concise Reply using Intelligent
Search Program (CRISP) aims to find a solution to optimize the difficulties
involved in searching details about tourism. CRISP is likely to serve as a
semantic web agent. Semantic Web Agents brings common formats for
integration and combination of data drawn from diverse sources. This brings
structure to the meaningful content of Web pages. It also creates an
environment where software agents roaming from page to page can readily
carry out sophisticated tasks for users.
CRISP comprises an information retriever and a converter. The Information retriever is responsible for retrieving information from Web Repository. It performs three major tasks namely

i) HTML to RDF conversion,

ii) Information integration and

iii) HTML generation.

Figure 3.1 Overall View of Web Page Representation System

CRISP is intended to generate semantic web content for tourism in Tamilnadu. The ultimate response generated by CRISP includes various details regarding the location the user searches for. CRISP could also provide the user with locations that are nearby to visit and the best time to visit those places based on various conditions. It also provides the user with details about the seasonal carnivals and festivals.

CRISP extracts and organises the relevant details from various web pages and gives it as response to the user. The user gets concrete replies for
his search instead of extensive list of irrelevant links. User will be able to search and arrange an efficient travel plan. Figure 3.1 shows the overall description of CRISP. End user searches for a tourist spot using a keyword or phrase. The response provided by CRISP will be in HTML containing all information about the location the user searches for.

### 3.3 WEB PAGE REPRESENTATION SYSTEM DESIGN

The components of CRISP encompasses three blocks namely information retriever, RDF converter and storage, and Integrator and HTML generator. When the user searches for the keyword /phrase, CRISP retrieves a few web pages using information retriever and converts it to machine understandable form using RDF converter. CRISP extracts the relevant details from various RDF documents and integrates it into a single document.

![Figure 3.2 Web Page Representation System Design](image-url)
Figure 3.2 shows the web page representation system design. Lastly, it converts the integrated document to HTML page and gives it as response to the user using Integrator and HTML generator. The implementation of the system is explained in the following sections.

3.4 INFORMATION RETREIVER

The information retriever block is responsible for data acquisition in the form of HTML pages. The user provides a keyword or phrase relating to a tourist place as input to CRISP’s search engine. Figure 3.3 shows the home page of the system. CRISP depends on an external server (Google) to probe for information and obtains HTML links. CRISP obtains first few links from Google’s search results. The HTML content corresponding to each link is retrieved and stored as files.

CRISP home page prompts the user to enter for a tourist spot. When the user types the tourist spot, a list of available locations matching the first letter will appear. The user has the freedom to select an option from the list or enter a keyword/phrase of their interest. CRISP maintains a database of the tourist locations in TamilNadu to avoid the user from searching for details other than tourist locations and checks the user search input with it. This is done because CRISP relies on Google to retrieve links, so it can possibly retrieve links for any input.

The valid search input is obtained and passed to a java program that is responsible for obtaining links and storing HTML files. The program establishes a HTTP connection with Google along with the concatenated query. Google’s search result page is examined to search for the anchor tag with href attribute in order to obtain first ten links. The links are extracted and HTTP connection is established with each link to obtain the HTML content of
each link. This result in storage of HTML files corresponding to the links extracted.

Figure 3.3 Home Page of CRISP

3.5 RDF CONVERTER AND STORAGE

The RDF converter paradigm transforms the HTML pages into RDF documents. It generates navigable structures that orient the users in current local context. Thus RDF provides a concrete syntax for the purpose of creating and exchanging metadata thereby enhancing the search results. XML is a method of storing RDF in a file. The specification of RDF uses XML as its interchange syntax. The resultant RDF files are stored for integration.

HTML Parser is a Java library used to parse HTML in either a linear or nested fashion. It is primarily used for transformation or extraction. It is a fast, robust and well tested package that is used to filter the tags in a HTML file and extract the content of a web page. Figure 3.4 is one of the
HTML file content for a web page that is satisfies the user query. This file is parsed and the tags and content are extracted.

![HTML File](image)

Figure 3.4 HTML File

The text content extracted from the HTML file is stored in a dynamic array. Initially the `<RDF>` tag which points to its default namespace is generated. The `<RDF>` tag encompasses a `<Description>` tag that consists of the name of the URL for which the RDF file is created.

The elements of Vector are compared with certain keywords and if a match is found, that particular sentence is enclosed within its corresponding tag. These tags are enclosed within the `<Description>` tag. Table 3.1 shows the list of keywords to be searched in the extracted string and the
corresponding RDF tags to be inserted. Figure 3.5 shows the equivalent RDF file.

The storage requirement for the RDF converter and storage during the transformation depends on the initial html file, RDF converted file and the final output HTML file. Intermediate files are created for processing. The storage requirement depends on the size of the HTML file that satisfies the user query, the size of the transformed RDF file and the size of the final HTML file. The file is parsed and the tags are then extracted. The content between the extracted tags are stored in a dynamic arrays. The size and the type of the array is to be considered. A new namespace for RDF is generated and the tags are enclosed with the <description> tag.

Table 3.1 RDF Tags

<table>
<thead>
<tr>
<th>Keywords To Search In Html Page</th>
<th>Equivalent Rdf Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>District</td>
<td>&lt;District&gt;</td>
</tr>
<tr>
<td>State</td>
<td>&lt;State&gt;</td>
</tr>
<tr>
<td>Temple</td>
<td>&lt;Temple&gt;</td>
</tr>
<tr>
<td>Tomb</td>
<td>&lt;Tomb&gt;</td>
</tr>
<tr>
<td>Estate</td>
<td>&lt;Estate&gt;</td>
</tr>
<tr>
<td>Dam/river/lake/falls/beach/sea</td>
<td>&lt;Water_Resources&gt;</td>
</tr>
<tr>
<td>Hill/mountain/peak/nature</td>
<td>&lt;Hill&gt;</td>
</tr>
<tr>
<td>Garden</td>
<td>&lt;Garden&gt;</td>
</tr>
<tr>
<td>Hotel/resort/restaurant</td>
<td>&lt;Hotel&gt;</td>
</tr>
<tr>
<td>Museum/palace/monument</td>
<td>&lt;Historical-Places&gt;</td>
</tr>
<tr>
<td>Festival/carnival</td>
<td>&lt;Festival&gt;</td>
</tr>
<tr>
<td>Zoo/sanctuary</td>
<td>&lt;Wild-Life&gt;</td>
</tr>
<tr>
<td>Airlines/transport/train/bus/road</td>
<td>&lt;Transport&gt;</td>
</tr>
<tr>
<td>Cave/rock/flora/fauna/ Forest</td>
<td>&lt;Forest&gt;</td>
</tr>
</tbody>
</table>
CRISP will perform further refinement on the stored RDF files to retrieve clear cut information and integrate that information into a single HTML file. The content is taken based on the tags such as hill, state, temple etc from all the RDF files. For example consider `<Hill>…</Hill>` tag, the content present within this tag is extracted from all the RDF files and this information is written into a HTML file along with its equivalent HTML tag.

This HTML file is provided to the user as final output which is in a table format. Figure 3.6 shows the final output. It provides concise report for the query given by the user. It contains the information and the corresponding link from which the information has been taken. The compared user query results is given in Table 3.2. Figure 3.7 gives the comparative results of the system with the existing search engine.
Consider an example, where a person, a backpacker wants to visit some of the famous tourist spots in Tamil Nadu. After some basic research, he decides to find out more tourism details about the locations given in Table 3.2. He first tries to Google these locations, which returns results that has several thousand links, all containing the keyword that was given as the search query.

But among these results only the first 2 to 3 pages are actually ‘unique content’ ‘related’ to the ‘desired’ location. All others are either duplicate versions of an original article or advertisements or local information not pertaining to tourism (like corporation details, government offices, retail shops, local business information or spam). And even the ones related to tourism are not very concise about the given query, they may simply mention a trivial fact about the keyword in context with the page’s content. e.g; a page called ‘List of Capital Cities’ may have only the following “Chennai is Tamil Nadu’s capital and has an international harbor, Hyderabad is the ….. and so on”. These non relevant results can be identified only after navigating to that page from the list of results in Google and then understanding it manually.

In the same scenario, the same Person will provide the search query in a semantic search engine (CRISP). The RDF repositories are scanned and all the relevant RDF descriptions are extracted from them. This list of domains are then presented to the Person, who chooses ‘tourism’ as the required domain. After this, CRISP engine collects all the relevant information under these RDF tags and consolidates them, along with their source URI, into a single page which is converted back to HTML and presented as a concise reply to Person in the familiar HTML format.

This example scenario is simulated for a few more locations/queries in both Google and CRISP and the results are analyzed for the percentage of tourism related information that is returned by both search engines. The
comparison for the query results are given in Table 3.2 and the graphical representation in the form of a chapter is given in Figure 3.7. These statistics clearly show that CRISP is marginally better than Google when it comes to meaningful search of a query.

<table>
<thead>
<tr>
<th>Search Query (name of the place)</th>
<th>% of tourism relevant data in web page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Using Google</td>
</tr>
<tr>
<td>Ooty</td>
<td>71</td>
</tr>
<tr>
<td>Yercaud</td>
<td>76</td>
</tr>
<tr>
<td>Chennai</td>
<td>61</td>
</tr>
<tr>
<td>Madurai</td>
<td>65</td>
</tr>
<tr>
<td>Kovai</td>
<td>72</td>
</tr>
</tbody>
</table>

**Figure 3.6 Concise Output for User Query**

**Table 3.2 Comparison of User Query Results**
3.7 SUMMARY

CRISP uses semantic web that provides richer interconnectivity among all objects (or content); thereby allows extracting data from multiple sources of information to discover new meaning and presents it to user. It enables the processing of metadata by RDF which provides a standard framework for expressing information about resources (metadata). The purpose of RDF is to create syntax to capture rich metadata and relationships. It can store, exchange and use machine readable information distributed throughout the web and provides a conceptual structure for presenting information to the user. Hence the final output is a concise document instead of links.

CRISP primary intention is for automation and integration of data from various sources. A flexible Web system is build which performs information search on heterogeneous data in RDF format and integrate them to provide a precise response in HTML format for TamilNadu tourism. CRISP extracts and organizes the relevant details from various web pages and
gives it as response to the user. The user gets concrete replies for his search instead of extensive list of irrelevant links.

The ultimate response generated by CRISP includes various details regarding the location/tourist spot the user searches for such as hill stations, temples, monuments, gardens, zoo, etc as a summary. The user query executed collects all the information from the retrieved pages and stores as RDF in the repository.

Some of the issues faced in the implementation of the system are that it depends on an already existing search engine (Google) to retrieve information from web. The HTML pages to be retrieved from different links might be from different servers so the retrieval may take some time. From the search results obtained, only a few links are taken for further processing due to certain limitations. The resultant paragraph is to be refined to provide an appropriate structure to the contents.

Some of the issues faced in the implementation of the system are:

- It depends on an already existing search engine (Google) to retrieve information from web.

\[ T_1 = (T_G) + (T_M) \]

Where \( T_G \) = Google’s Search Time and

\[ T_M = T_{REQ} + T_{RESP} \]

Message request and response time between CRISP and Google

- The HTML pages to be retrieved from different links might be from different servers so the retrieval may take some time.

\[ T_2 = \sum_{s=1}^{N} [T_R(S)] \]
Where \( T_R \) = Time for retrieving content from Server ‘S’

\[ N = \text{number of unique server IPs in the Google results page} \]

\[ S = \text{Server/Iteration Number} \]

- From the search results obtained, only a few links are taken for further processing due to certain limitations.

Though RDF enhances the efficiency of searching, complex querying reasoning is not possible and there is a need for ontological representation which is also discussed by Patel Schneider (2002). The stored RDF documents can act as a repository for creating a semantic web search engine, but the space occupancy will be huge and is impossible to do for millions of pages. There is a possibility of redundant HTML file processing and storage. Hence there is a need to annotate or include additional information in the existing web pages.